
# Distributed Systems

# Global Organization

- **Introduction**
- **Distributed programing & principles**
  - Communication & Client/Server model (Sockets : TCP – UDP – Multicast)
  - Processes & Threading
  - Distributed Object Model
    - RMI + Servlets (JSP) + Applets
    - CORBA
    - EJB
  - Message Oriented Middleware (MOM) – JMS
  - Web Services + SOA
- **Distributed algorithms**
  - Logical Time & Global State
  - Coordination & Agreement
- **Distributed Disgens**
  - Replication
  - Distributed Transactions
  - Shared memory Interprocesses – JavaSpaces
  - Google

# Plan

- Definition
- Middleware
- Distribution
- DS Types
- DS Goals
- DS Chaleneges

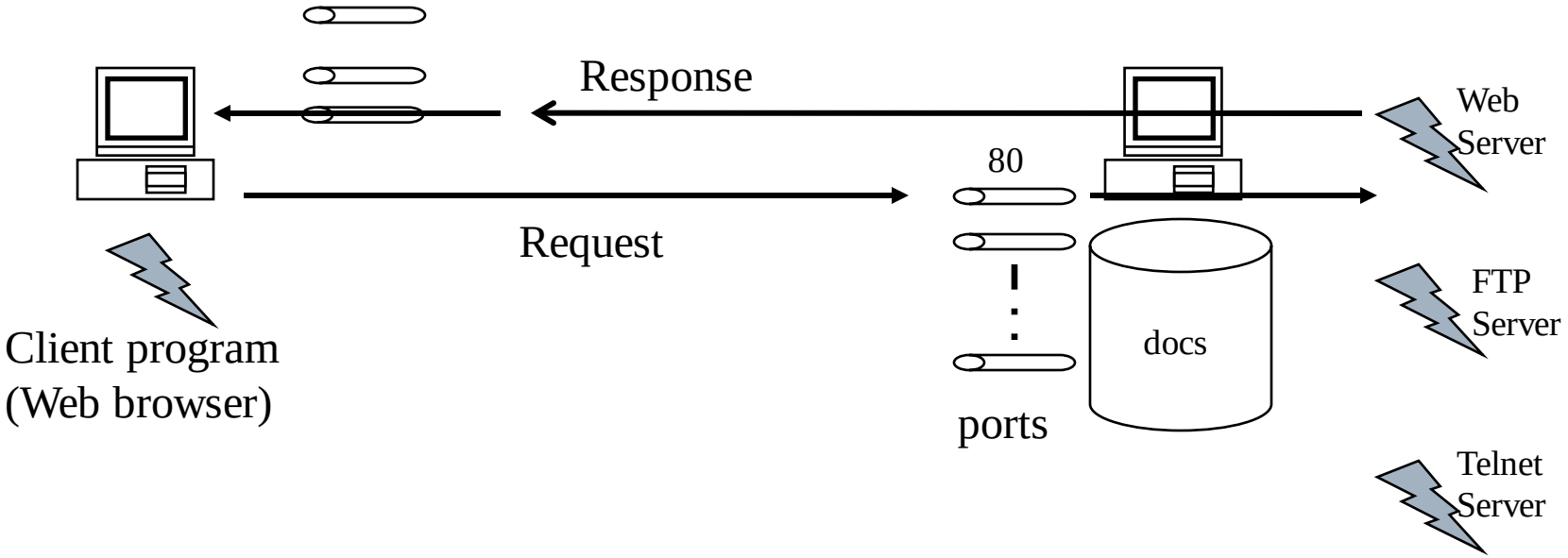
# Plan

- Definition
- Middleware
- Distribution
- DS Types
- DS Goals
- DS Chaleneges

# Definition

- نظام (تطبيق) موزع
  - تعاون = مجموعة من الفعاليات (Processes) التي تدور في عدة مواقع متصلة فيما بينها بواسطة شبكة اتصال و تتعاون فيما بينها لتحقيق هدف واحد.
- A distributed system = collection of independent computers that appears to its users as a single coherent system.
  - اتصال بالرسائل =
    - الرسالة = مجموعة من البايتات الممررة بين أجزاء النظام الموزع
    - برتوكول الاتصال = مجموعة من الرسائل + قواعد لتبادل هذه الرسائل + الأفعال الواجب اتخاذها عند استقبال الرسائل

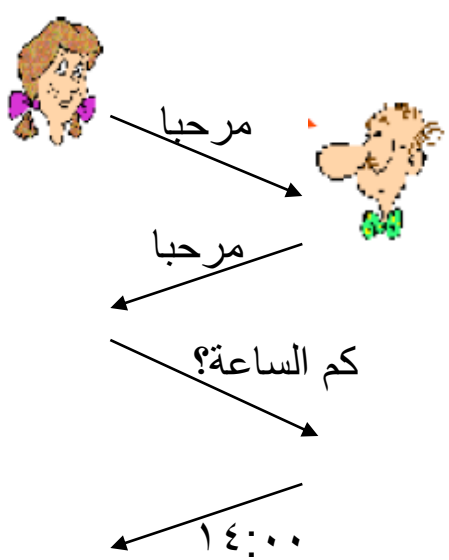
# Client\Server Architecture

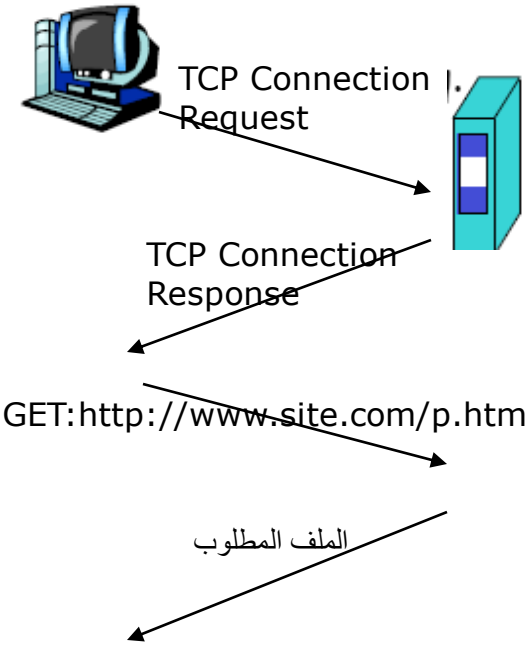



Available ports = 1024 → $2^{16} - 1$

# Communication Protocols

- البرتوكول = صيغة وترتيب الرسائل المرسلة / المستقبلة عبر مكونات الشبكة و الأفعال الواجب اتخاذها عند استقبال الرسائل



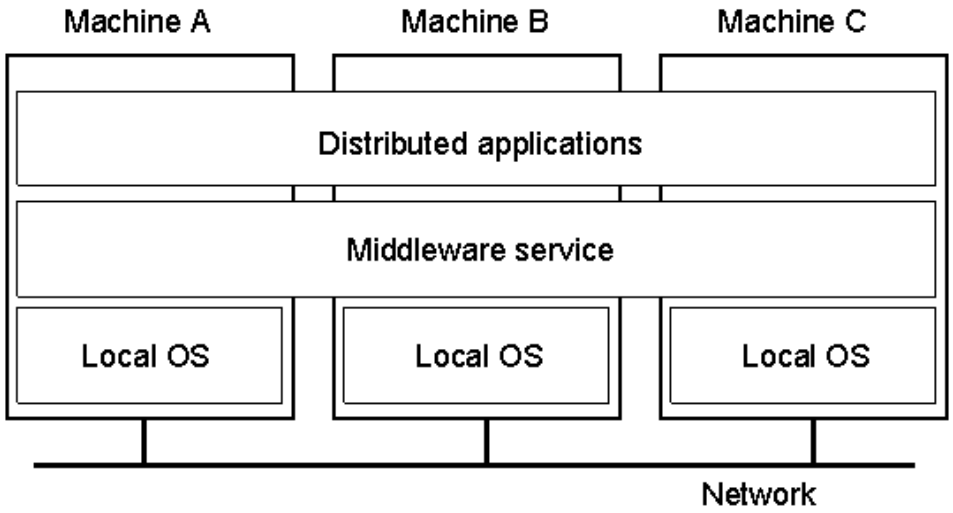




- Connection-Oriented Protocols
  - TCP
  - Secure communication protocols
- Connection-less Protocols
  - UDP
  - Group communication protocols

# Plan

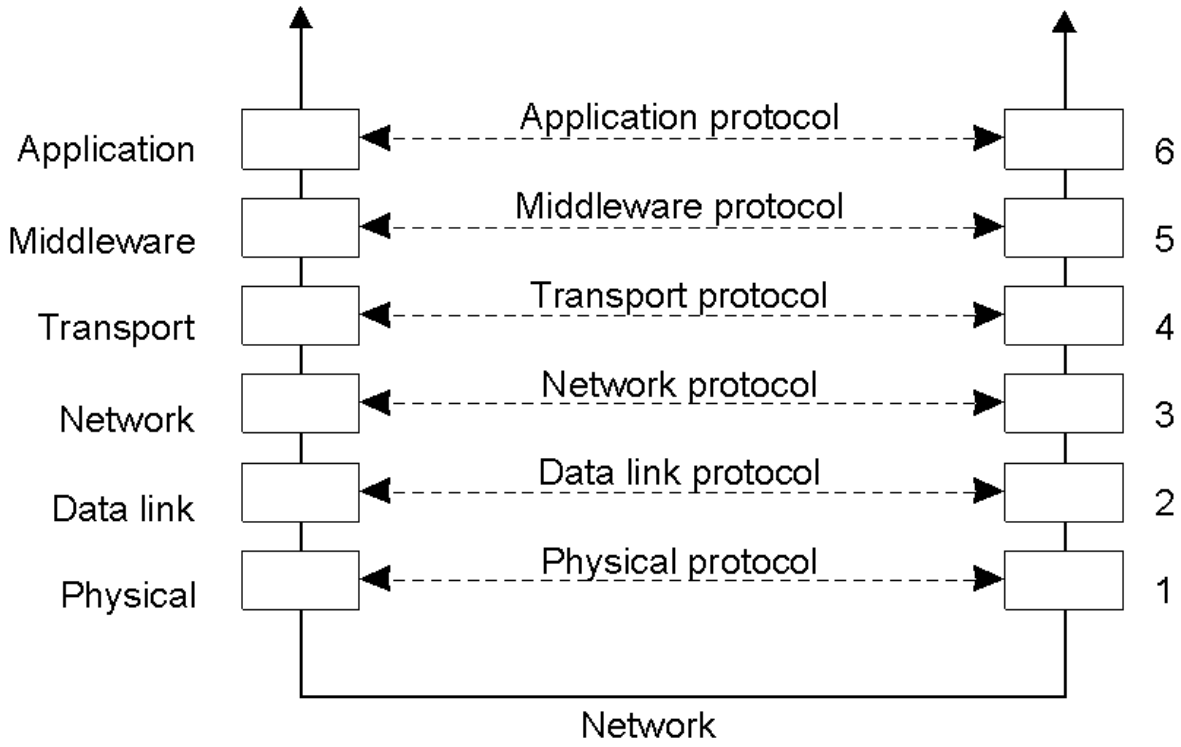
- Definition
- Middleware
- Distribution
- DS Types
- DS Goals
- DS Chaleneges

# Middleware




الـ Middleware= هو مجموعة من البرمجيات الموزعة على كل آلة في النظام و مهمتها تقديم واجهة للاتصال و التخاطب لأجزاء النظام التي تعمل على هذه الآلات.
كما أنه يخفي تفاصيل الاتصال عبر الشبكة عن التطبيق الموزع.

# Middleware


Application
Middleware
Transport
Network
Data link
Physical
Application protocol
Middleware protocol
Transport protocol
Network protocol
Data link protocol
Physical protocol
6
5
4
3
2
1
Network

# طبقات شبكة الانترنيت

- Application = مجموعة التطبيقات المدعومة من الشبكة
  - HTTP, SMTP, FTP, Telnet, ...
- Middleware = البرمجيات الوسيطة و التي تدعم بناء تطبيقات موزعة و تخفي تفاصيل الاتصال على الشبكة
  - RMI, CORBA, DCOM, ..
- Transport = التحويل من نقطة لنقطة
  - TCP, UDP, …
- Network = تحويل الرزم من الـ Source إلى الـ Destination
  - IP, Routing Protocol
- Data Link = تحويل المعطيات بين النقاط المتجاورة و اكتشاف الأخطاء
  - PPP, Ethernet
- Physical = البيتات على الكابل (مستوى الجهد للـ ٠ و ١ / معدل إرسال البيتات / الاتصال بالاتجاهين أو لا .....)

# Plan

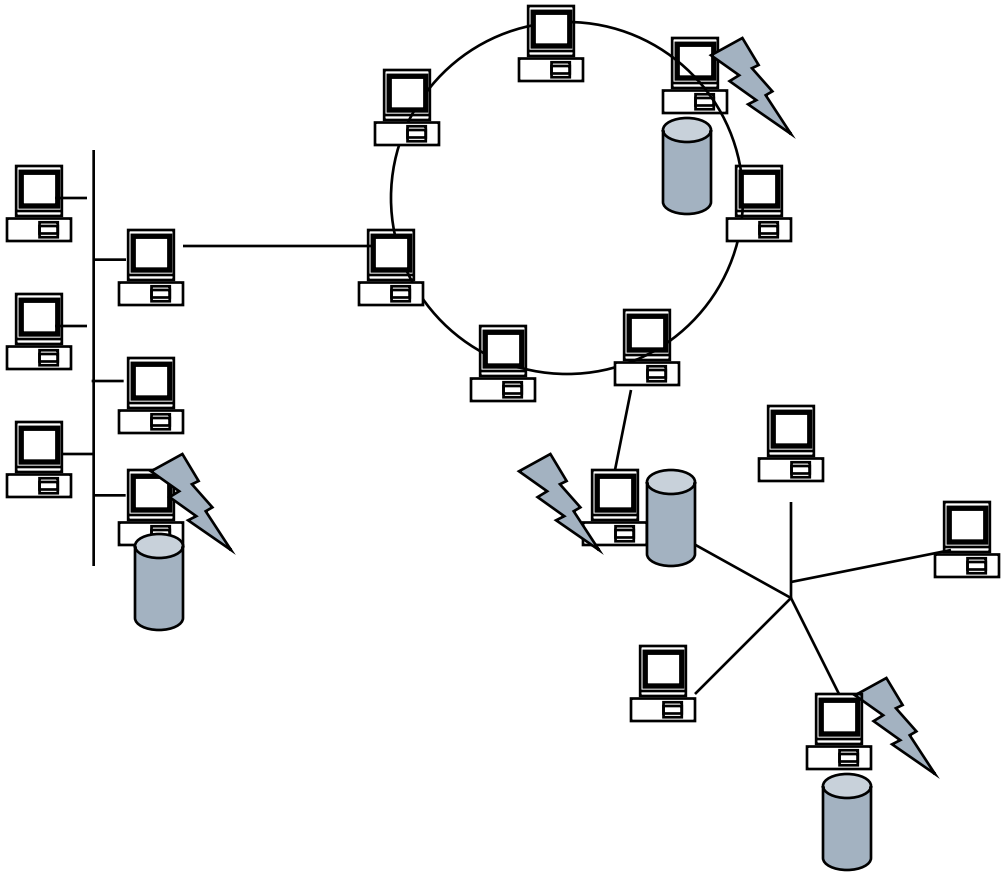
- Definition
- Middleware
- Distribution
- DS Types
- DS Goals
- DS Chaleneges

# Distribution

- Distributed system = spiliting the system in several (smaller) parts and spreading those parts across the network computers
- Examples :
  - **WEB** as big system of information
  - **DNS** (Domain Name Server)
  - **E-MAIL** System
  - **ROUTING** in the network

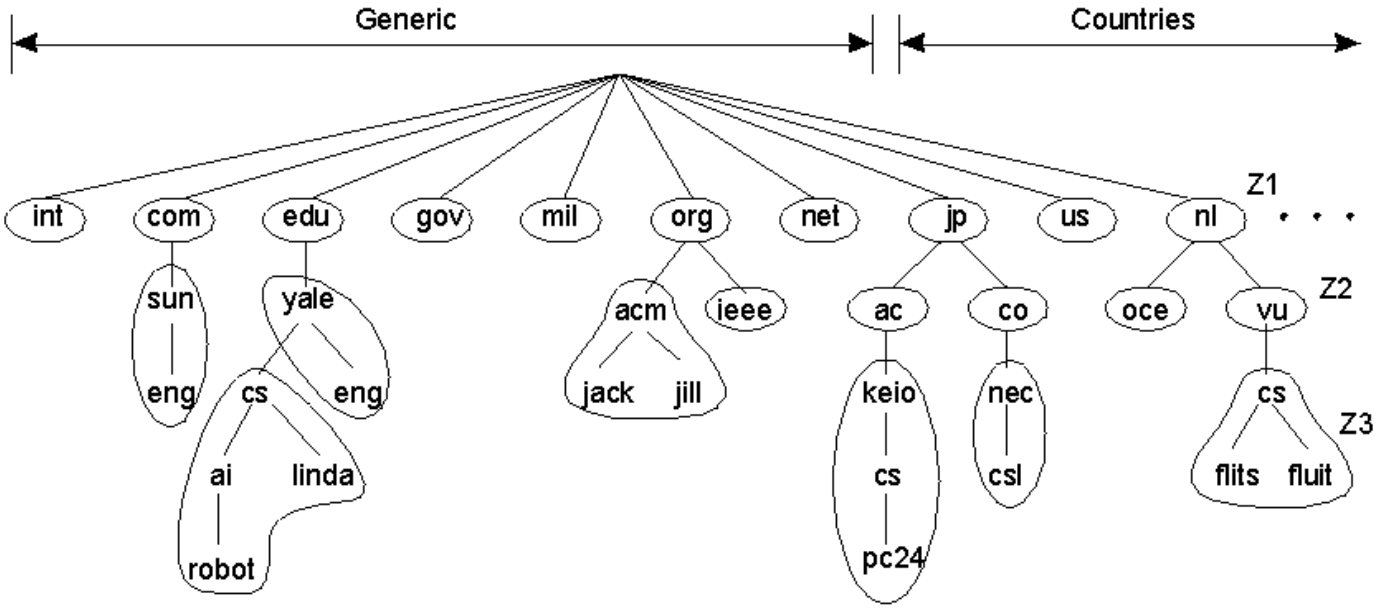
- لماذا نوزع الأنظمة :
  - تحسين في أداء النظام + توفير في التكاليف ( مثال : Google, Grid Computing)
  - في حال تعطل عقدة في النظام يبقى النظام في حالة عمل ( مثال : DNS)
  - قابلية النظام للتوسع (بعدد المستخدمين و الموارد) بسهولة ( مثال : Skybe)
  - بعض المسائل بالأصل موزعة جغرافياً و على عدة آلات (مثال : موقع E-Com. و مستودعاته - المصرف)
- السيئات :
  - القرارات يتم اتخاذها استناداً على المعلومات المحلية و ليس بحسب حالة النظام العامة

# Distribution / WEB



# Distribution / DNS

- Hierarchic Structure
- Hierarchic Algorithm



# Distribution / DNS

- DNS = قاعدة معطيات موزعة و محققة كهرمية من Name Servers
  - الوظيفة = تحويل اسم الحاسب البعيد إلى IP حتى نتمكن من تسيير الـ Datagrams
  - كل مجال له local Name Server
  - يتم توجيه طلبات الـ DNS نحو المخدم المحلي أولاً
  - في حال فشل الـ local DNS ينتقل الطلب نحو الـ Root DNS
  - لدينا نوعين من الطلبات Iterative/Recursive Request
- لماذا DNS ليس Centralized؟
  - نقطة حساسة (أعطال)
  - حجم العبور و الاتصال إلى هذه العقدة المركزية
  - لا سهولة بالتوسع
  - الصيانة
  - قاعدة معطيات مركزية و بعيدة

# Plan

- Definition
- Middleware
- Distribution
- DS Types
- DS Goals
- DS Chaleneges

# Distriuted System Goals

- Connecting users & Resource Sharing
- Scalability
- Transparency
- Openness
- Concurrency

# Resource Sharing

- The main goal of a DS is to make it easy for users to access remote resources, and share them with other users in a controlled way.
- Computer resources :
  - hardware resources (CPU, disks, printers, memory, network capacity, ..)
  - Data/software resources (compiler, programming library, files, databases, objects, .. services)

# Scalability

- System is Scalabale = if it remain effective when nb of users/recourses increase
- Challenges :
  - Performance loss & bottleneck
    - Hierarchic algo as in DNS <> linear algo
  - Software resources running out
    - Ex : IPs
      - NAT (Network Adress Translation)
      - IP6 instead of IP4 ➔ modif to many soft components
  - Cost of physical resources <> users nb

# Scalability

- Problem = performance
- solutions
  - Hiding Communication
    - Asynchronous communications
      - multithreaded client & replicated multithreaded server
      - MOM
      - ...
    - Code mobile
      - Applets for checking **form** values
  - Replication & Caching
    - ➔ Consistency problems + concurrency updates
  - Distribution

# Openness

- Open DSs can be built from heterogeneous hardware and software, and from different vendors.
- Open DS can communicate with any other open systems by standard rules (message format, content and meaning)
- Solution = specification + publication of service interfaces as standards
  - Ex : OMG **specify** & **publish** CORBA interfaces + documentation to be implemented by several vendors.
  - Ex : WEB (HTML, URL, HTTP) = several types of browsers from several vendors.

# Concurrency (Example 1)

- Clients access the shared resource at the same time
  - Increase throughput (good)
  - Conflict may produce inconsistent result (bad)
- So, client operations must be synchronized to mainain the data consistent

| **Transaction *T*** | | **Transaction *U*** | |
|---|---|---|---|
| *balance = b.getBalance();*<br>*b.setBalance(balance*1.1);*<br>*a.withdraw(balance/10)* | | *balance = b.getBalance();*<br>*b.setBalance(balance*1.1);*<br>*c.withdraw(balance/10)* | |
| *balance = b.getBalance();* | $200 | | |
| | | *balance = b.getBalance();* | $200 |
| | | *b.setBalance(balance*1.1);* | $220 |
| *b.setBalance(balance*1.1);* | $220 | | |
| *a.withdraw(balance/10)* | $80 | | |
| | | *c.withdraw(balance/10)* | $280 |

Lost update problem

# Concurrency (Example 1)

| Transaction *V*: | | Transaction *W*: | |
|---|---|---|---|
| *a.withdraw(100)*<br>*b.deposit(100)* | | *aBranch.branchTotal()* | |
| *a.withdraw(100);* | $100 | | |
| | | *total = a.getBalance()* | $100 |
| | | *total = total+b.getBalance()* | $300 |
| | | *total = total+c.getBalance()* | |
| *b.deposit(100)* | $300 | ⋮ | |

Inconsistent retrivals problem

# Transparency (1/3)

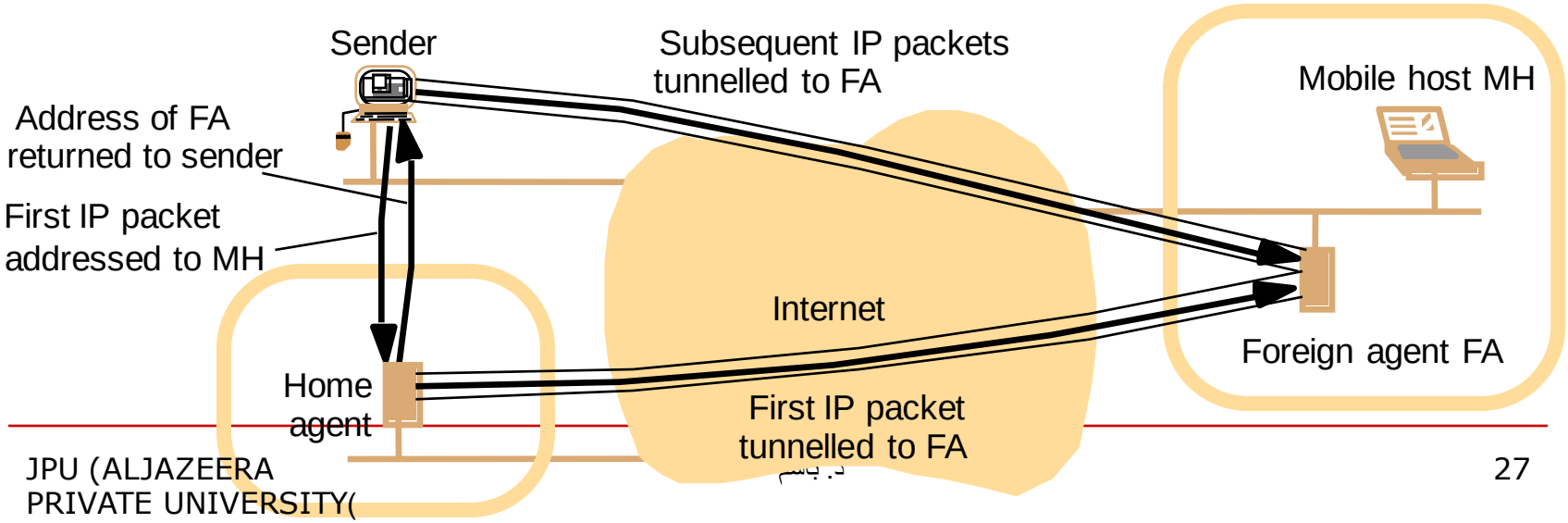
- Transparency = Hide certain aspects of distribution to the application programmer. Transparency forms :
  - **Access transparency**: enables local and remote resources to be accessed using identical operations.
    - Ex : graphical FTP
    - Ex : access to local objects & remote objects
  - **Location transparency**: enables resources to be accessed without knowledge of their physical or network location.
    - Ex: IP address
  - **Replication transparency**: enables multiple instances of resources to be used to increase reliability and performance without knowledge of the replicas by users or application programmers.
  - **Persistence transparency**: Hide whether a (software) resource is in memory or on disk.

# Transparency (2/3)

- **Concurrency transparency**: enables several processes to operate concurrently using shared resources without interference between them.
- **Failure transparency**: enables the concealment of faults, allowing users and application programs to complete their tasks despite the failure of hardware or software components.
- **Mobility transparency**: allows the movement of resources and clients within a system without affecting the operation of users or programs (migration of objects + relocation of resources)
- **Performance transparency**: allows the system to be reconfigured to improve performance as loads vary.
- **Scaling transparency**: allows the system and applications to expand in scale without change to the system structure or the application algorithms.

# Transparency (optimization)

- ملاحظة : من المفيد أحياناً تجاوز الـ Transparency و عمل نوع من الـ Optimization
  - مثال ١ = عدة أغراض بعيدة موجودة على نفس الحاسب. في هذه الحالة من الأفضل أن يتفاعلوا مع بعضهم البعض مباشرة و توفير المرور بطبقة الـ Middleware
  - مثال ٢ = شخص يتصل بموقع له Mobile IP. لنتخيل الحالة عندما يتواجدا جغرافياً بقرب بعضهما البعض ـ في هذه الحالة قد تذهب الرزم من أول الأرض لآخرها مرتين حتى تصل بين نقطتين بالأصل بينهما مسافة من عدة أمتار.
  - ....



# Plan

- Definition
- Middleware
- Distribution
- DS Types
- DS Goals
- DS Chaleneges

# Distriuted System Challenges (1)

- Heterogeneity
  - Networks
  - Computer hardware
  - Operating Systems
  - Programming languages
  - Software prviders
- Security (Confidentiality – integrity – availability)
  - Solution = encryption techniques
  - Problems for today:
    - Denial of service attacks
    - Security of mobile code

# Distriuted System Challenges (2)

- Network Failure
  - Lose messages (solution = Retransmition)
  - Lose data (solution = Data Replication)
  - Lose link (solution = route redundancy)

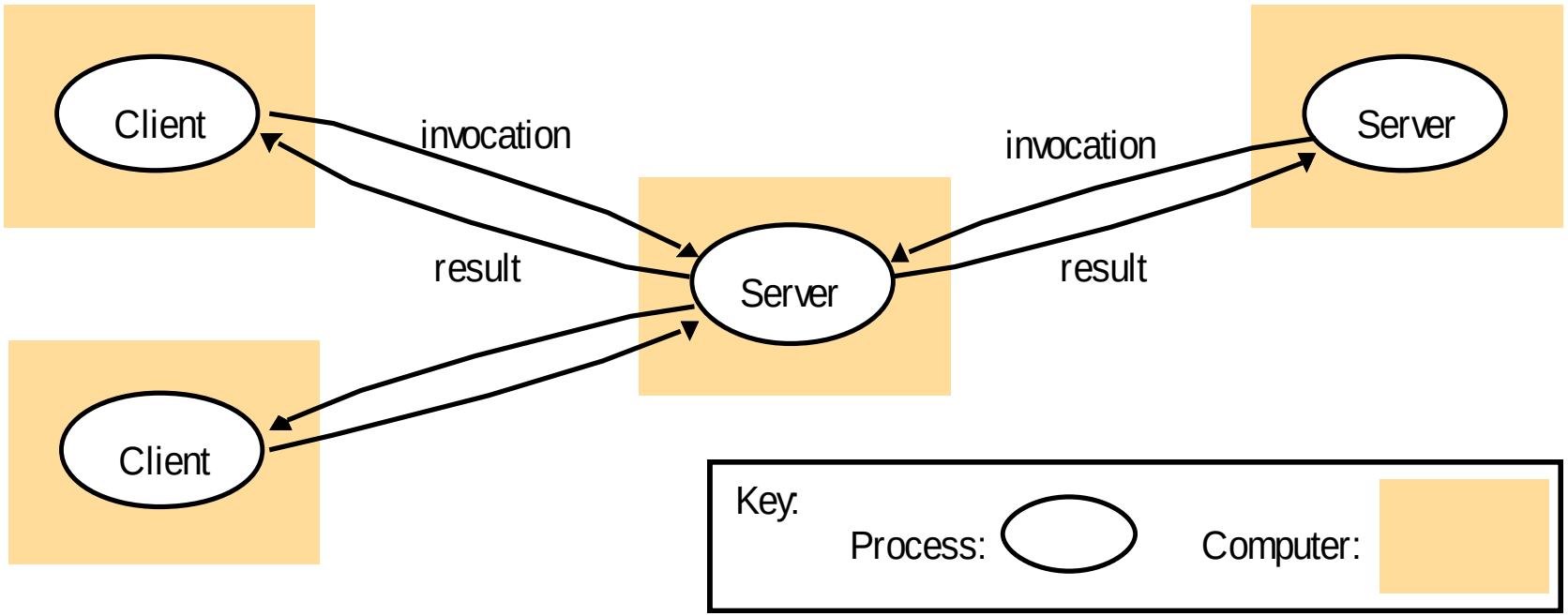
# Plan

- Definition
- Middleware
- Distribution
- DS Types
- DS Goals
- DS Chaleneges

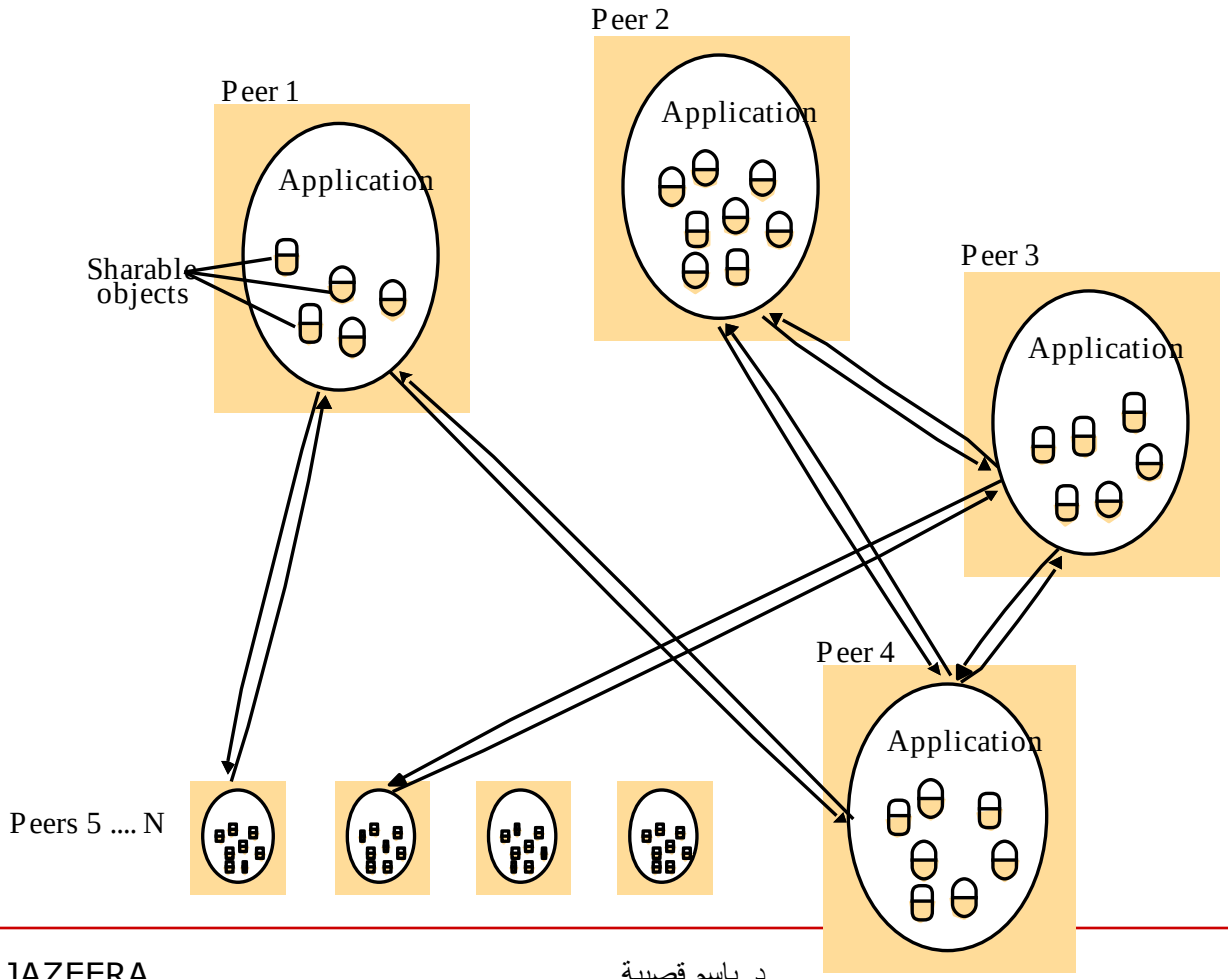
# Distributed System Types

- حسب البنية
  - Client/Server
  - Peer to Peer Systems
  - ...
- حسب نمط التفاعل
  - متزامن
  - لا متزامن
- حسب الهدف
  - أنظمة المعلومات الموزعة
    - أنظمة تدفق المعطيات الموزعة
    - أنظمة التعاون الموزعة
    - أنظمة تدفق الأحداث الموزعة
    - ....
  - أنظمة الحسابات الموزعة
  - ...
- ...

# Client/Server Systems



# P2P Systems


Peer 2
Peer 1
Application
Application
Sharable
objects
Peer 3
Application
Peer 4
Application
Peers 5 .... N

# أنظمة تدفق المعطيات الموزعة

- المسائل المهمة في هذا النوع من الأنظمة : التنسيق بين المهمات و تكييف المهمات حسب البيئة و إدارة مجموعات مختلفة من المستخدمين + الأمن + المناقلات الموزعة

- **موقع تجارة الكترونية**
  - الهدف = الربط بين خدمة موقع ويب (قائمة السلع) و خدمات المستودعات التي من الممكن أن تكون موزعة على عدة محافظات و خدمات الدفع الالكترونية
- **مصرف**
  - الهدف = الربط بين خدمة الزبائن (ويب أو ATM أو في المصرف) الموزعة أصلاً مع خدمة الحسابات المالية و التي هي أيضاً موزعة + تضاعف المعطيات و تحديثاتها (و التحديثات المتسايرة)

# أنظمة التعاون الموزعة

- **نظام تحرير موزع ( اتصالات لا متزامنة)**
  - تقسيم النظام إلى أجزاء
  - توزيع الحقوق على مستخدمي النظام
  - تحديد درجة العزل المناسبة للتطبيق (رؤية التغييرات من قبل المستخدم A الناتجة عن المستخدم B )
- **محاضرات عن بعد (اتصالات متزامنة)**
  - إدارة الجلسة (ارتباط ـ فك ارتباط)
  - إدارة الوصول للوثائق
  - إدارة التفاعل بين المشتركين
- **لعبة موزعة بالزمن الحقيقي**
- **....**

# الأنظمة حسب نمط التفاعل

- الأنظمة الموزعة المتزامنة
  - زمن تنفيذ كل عملية محصور بين قيم [min, max] محددة
  - كل رسالة على القناة تصل ضمن فترة زمنية محددة
  - ساعة كل فعالية بالنظام تنزاح عن الوقت الحقيقي بمجال محدد
- الأنظمة الموزعة اللامتزامنة
  - زمن تنفيذ كل عملية غير محدد
  - زمن إرسال الرسائل على القناة غير محدد
  - ساعة كل فعالية بالنظام تنزاح عشوائياً عن الوقت الحقيقي
- ملاحظة : لا نخلط مع الاتصالات المتزامنة و اللا متزامنة

# Conclusion

- DS' Consequences
  - Concurrency
  - No global clock
    - only communication by sending messages
  - Independent failures

Questions … ???